1 Introduction

2 File Names

2.1 File Suffixes

Map files by the name
Atlantis.map
Yorkshire.map
USA.map

Card.java
Desk.java
DeckTest.java
Hand.java
Driver.java
GameEngine.java
ICharacter.java
PlayerModel.java
MapModel.java
ModelException.java

2.2 Common File Names

File name use
Package-info.java
README
LICENSE

3 File Organization

3.1 Java Source Files

3.1.1 Beginning Comments

```
package risk.game;
package risk.model.character;
package risk.ui;
package risk.model;

/**
 * @author hcanta
 *
 */
/**
 * @author akif
 *
 */
```

3.1.2 Package and Import Statements

```
import java.awt.BorderLayout;
import java.awt.Canvas;
import java.awt.Dimension;

import javax.swing.JFrame;
import javax.swing.JMenuBar;

import org.apache.log4j.Logger;

import risk.model.character.PlayerModel;
```

```
import java.util.ArrayList;
import java.util.Collections;

import risk.model.ModelException;
import risk.model.character.ICharacter;
import risk.model.character.PlayerModel;
import risk.util.RiskEnum.RiskColor;
import risk.util.map.RiskBoard;
```

3.1.3 <u>Class and Interface Declarations</u>

```
public class Driver {

        /**
         * Main Method of the class that creates the Game Instance and starts the
         * game.
         *
         * @param new_args
         *          contains the supplied command-line arguments as an array of
         *          String objects
         */
public class PlayerModel extends Observable implements ICharacter

/**
         * Constructor that initializes default values
         */
        public PlayerModel()

/**
         * Constructor that assigns its parameter to class attributes
         *
         * @param new_playerName
         *          Player name
         */
```

4 <u>Indentation</u>

Four spaces were used as unit of indentation

4.1 <u>Line Length</u>

The lines are not of length greater than 50-60 characters so that it could be easily handled by terminals

4.2 <u>Wrapping Lines</u>

The lines which could not fit as single lines have been broken after a comma or before an operator

```
if(this.territoriesOwned.contains(territory1.toLowerCase().trim()) &&
                                        this.territoriesOwned.contains(territory2.toLowerCase().trim()))
```

5 <u>Comments</u>

Both implementation comments and documentation comments has been used

5.1 <u>Implementation Comment Formats</u>

5.1.1 <u>Block Comments</u>

```
/*
 * Generates computer player
 */

/*
 * Give the armies according to the risk rules
 */
```

5.1.2 <u>Single-Line Comments</u>

```
/* Default constructor */

/* Constructor that initializes default values */
```

5.1.3 <u>Trailing Comments</u>

Some very short comments has also been used along the lines.

5.1.4 <u>End-Of-Line Comments</u>

```
// TODO Auto-generated method stub
//Game.getInstance().start();
// TODO Auto-generated catch block
// default constructor
```

5.2 <u>Documentation Comments</u>

```
/**
 * Constructor that assigns its parameter to class attributes
 *
 * @param new_playerName
 *            Player name
 */


/**
 * @param nbArmiesToBePlaced the nbArmiesToBePlaced to set
 */

/**
 *
 * @param playername The player name
 * @param territory1 The origin territory
 * @param territory2 The destination territory
 * @param armies The number of armies to be moved
 * @throws Model Exception
 */
```

6 Declarations

6.1 Number Per Line

In most of the source code One declaration per line was done

6.2 Initialization

The local variables were initialized were they were declared

6.3 Placement

Declarations were mostly at the beginning of the block

6.4 Class and Interface Declarations

```
public class GameEngine
                          {
public GameEngine ()
                         {
public void setNumberofPlayers(short numberPl)
            {
```

7 Statements

7.1 Simple Statements

Every line has one statement to avoid confusion

7.2 Compound Statements

List of statements were enclosed within braces { }

7.3 return Statements

```
return mapWidth;
```

7.4 if, if-else, if else-if else Statements

The statements are with braces and well placed

```
if(this.territoriesOwned.contains(territory1.toLowerCase().trim()) &&
                                          this.territoriesOwned.contains(territory2.toLowerCase().trim()))
                        {
                                    int armyOn1 = RiskBoard.Instance.getTerritory(territory1).getArmyOn();
                                    int armyOn2 = RiskBoard.Instance.getTerritory(territory2).getArmyOn();
                                    if( armyOn1 > armies)
                                    {
                                                RiskBoard.Instance.getTerritory(territory1).setArmyOn(armyOn1 - armies);
                                                RiskBoard.Instance.getTerritory(territory2).setArmyOn(armyOn2 + armies);
                                    }
                                    else
                                    {
                                                throw new ModelException("The amount of armies to be moved exceed the
amount of amries present");
                                    }

                        }
                        else
```

```
                {
                        throw new ModelException("One or more of the territory is not owned by the player");
                }
```

7.5 <u>for Statements</u>

Are of the form

```
for (initialization; condition; update) {
    statements;
}
```

7.6 <u>while Statements</u>

7.7 <u>do-while Statements</u>

7.8 <u>switch Statements</u>

7.9 <u>try-catch Statements</u>

```
try {
                        thread.join(0); // stops the thread
                        System.out.println("Game loop is stopped.");
                } catch (Exception e) {
                        e.printStackTrace();
                        System.out.println("EXCEPTION HERE void stop function.");
                }
        }
```

8 <u>White Space</u>

8.1 <u>Blank Lines</u>

Blank lines were left between sections of the code and between class interface declarations

8.2 <u>Blank Spaces</u>

Spaces were left after keywords to avoid confusion

9 <u>Naming Conventions</u>

Packages:- risk.ui; package risk.game;

Classes :- class GameEngine public class ModelException extends RuntimeException

Methods :- public void createBots() public void addHumanPlayer(String name)

Variables :- short i = 1 int nbArmiesToBePlaced = 0; int player = i%pl.size();

Constants :- public static final int WINDOW_WIDTH = 840; private static final long serialVersionUID = -228084634482818388L;

10 <u>Programming Practices</u>

10.1 <u>Providing Access to Instance and Class Variables</u>

No unnecessary public access to class variables or instances

```
        private ArrayList<ICharacter> players;
```

```
private short numberOfPlayers;
```

10.2 Referring to Class Variables and Methods

```
Collections.shuffle(countries);

pl.get(player).addTerritory(countries.get(i));
pl.get(player).decrementArmies();
RiskBoard.Instance.getTerritory(countries.get(i)).setOwnerID(pl.get(player).getTurnID());
RiskBoard.Instance.getTerritory(countries.get(i)).setArmyOn(1);
```

10.3 Constants

10.4 Variable Assignments

Several variables are not assigned to the same value so as to increase readability

10.5 Miscellaneous Practices

10.5.1 Parentheses

```
if(players.get(i).getName().equalsIgnoreCase(playername))

if(players.get(i).getTurnID()==(short)RiskBoard.Instance.getContinent(continent).getOwnerID())
```

10.5.2 Returning Values

```
return mapWidth;
```